UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

TIMELINE WEB BROWSER E ARQUITETURA DE SOFTWARE PARA APLICAÇÕES WEB 2.0

TRABALHO DE GRADUAÇÃO

Aluno: Livar Correia de Oliveira Cavalcanti Cunha (lcocc@cin.ufpe.br)
Orientador: Prof. Silvio Lemos Meira (srlm@cin.ufpe.br)

24 de Setembro de 2006

ASSINATURAS

Este Trabalho de Graduação é resultado dos esforços do aluno Livar Correia de Oliveira Cavalcanti Cunha, sob a orientação do professor Silvio Lemos Meira, sob o título de "*Timeline Web Browser e Arquitetura para Aplicações Web 2.0*". Todos abaixo estão de acordo com o conteúdo deste documento e os resultados deste Trabalho de Graduação.

_____

Livar Correia de Oliveira Calvancanti Cunha

_____

Silvio Lemos Meira

"Time is on my side. Yes it is!"
Time is on my side

(Rolling Stones – 12X5)

Agradecimentos

Este trabalho marca o fim de mais uma fase na minha vida. A jornada até aqui teria sido mais difícil se tivesse percorrido este caminho sozinho. Eu sempre estive, e estarei bem acompanhado. É impossível agradecer a todas as pessoas que me ajudaram a chegar até aqui e que vão me ajudar a chegar ainda mais longe, portanto gostaria de agradecer primeiro as pessoas que não couberem ou que eu me esquecer de agradecer nas próximas linhas deste texto.

Gostaria de agradecer em segundo lugar minha família, os primeiros responsáveis por eu ser quem eu sou hoje. Com certeza foi a melhor criação possível.

Gostaria de agradecer também aos grandes professores que tive, todos eles, do começo do colégio até o fim da graduação.

Gostaria de agradecer a Allan e Rafael (Cabelo), pelas várias reuniões e discussões, que ajudaram a gerar a versão final deste documento. E também, Renan, Thiago, Fernando, José Carlos, e todos que foram meus amigos, mais próximos e mais afastados durante a graduação.

Por último, gostaria de agradecer a Katharina, que passou a me acompanhar nos últimos meses, mas sei que estes meses logo serão anos, e que desde já está fazendo sacrifícios pelo meu futuro profissional e acadêmico.

Gostaria de terminar este agradecimento como comecei, agradecendo as pessoas que eu não citei ou não referenciei aqui. A todos obrigado, e sei que ainda vou ter muito que agradecer a todos vocês.

Resumo

Este trabalho se propõe a analisar o conceito de Web 2.0 e de plataforma web e estabelecer uma arquitetura para aplicações baseadas neste conceito. É apresentada uma visão geral destes conceitos, e em seguida são abordadas as principais tecnologias usadas para desenvolvimento de aplicações na plataforma web. A seguir é sugerida uma arquitetura que use estas tecnologias. E por fim é descrita uma aplicação chamada de *Timeline Web Browser*, que se baseia no conceito de Web 2.0 e usa a arquitetura descrita neste trabalho.

Palavras chave: Web 2.0, Plataforma Web, Web Services, SOA, SOAP, Padrões de Projetos, Arquitetura de Software.

Abstract

This work intends to make an analysis of the Web 2.0 and Web Platform concepts and establish an architecture to applications based on this concept. It presents an overview of these concepts, and then describes the main technologies used on the development of web platform applications. Then it establishes an architecture that uses these technologies. At last, it describes an application called Timeline Web Browser, which is based on the concepts of Web 2.0 and web platform and uses the architecture described here.

Key Words: Web 2.0, Web Platform, Web Services, SOA, SOAP, Design Patterns, Software Architecture.

Índice

Índice de Figuras

Índice de Listagens

1. Introdução

"A long journey begins with the first step"
Neil Armstrong

Com uma alusão aos números de versões, normalmente usados para indicar novas versões de software, o termo "Web 2.0" indica uma forma aprimorada da *World Wide Web*. Esta nova web dá ênfase a ferramentas e plataformas que permitem ao usuário não apenas ser um receptor de informações, mas um produtor .

A Web 2.0 se diferencia de sua versão anterior (Web 1.0) à medida que ela se distancia dos sites estáticos, do uso de engenhos de busca e da navegação de um site para um outro, em direção a uma *web* mais dinâmica e interativa.

Atualmente, vários fatores contribuem para o avanço da web em direção a sua versão 2.0:

- A popularização da banda larga, proporcionando um uso cada vez maior da internet.
- O aumento de pessoas usando a internet para realizar atividades que antes eram realizadas de forma *offline*, como compras, leitura de notícias etc.
- A popularização de tecnologias como AJAX, que possibilitam ao usuário uma experiência de uso mais rica na web.

São exatamente tecnologias como AJAX que possibilitam a web "imitar" aplicações desktop como processadores de texto, planilhas de cálculo e programas de apresentação de slides. Além de sistemas operacionais que funcionam dentro de um browser. Não sistemas operacionais em si, mas sim, plataformas de aplicações. Serviços com características e aplicações similares a um ambiente desktop. A principal diferença é que eles podem ser usados a partir de qualquer browser.

Dando um passo adiante, os sites "Web 2.0" permitem que desenvolvedores externos construam novos serviços em cima de serviços providos por sites já existentes, semelhante à maneira como a Microsoft incentiva os programados a desenvolverem em cima do windows .

Os serviços destes sites “Web 2.0” são construídos e disponibilizados para os desenvolvedores em geral em cima de arquiteturas de web services que usam ferramentas de publicação para publicar suas APIs para a comunidade de desenvolvedores.

Isto leva a um importante conceito de “Web 2.0”: o da web como plataforma de serviços, onde aplicações web-based não deveriam oferecer suas funcionalidades apenas através do browser, mas também através de web services . Desta forma, transformando estas aplicações em provedores de serviços e a web em plataforma destes serviços, semelhante a um sistema operacional.

1.1.Objetivos

Este trabalho tem como objetivo realizar estudo do conceito de web 2.0 que coloca a internet como uma plataforma de desenvolvimento baseada em serviços, além da definição de uma arquitetura para aplicações baseadas neste conceito. Estes serviços são disponibilizados através de web services, que são usados por outras aplicações para prover novas funcionalidades ainda mais complexas. O trabalho possui dois objetivos principais, listados abaixo:

1. Definição de uma arquitetura para aplicações que se baseiam no conceito de que a internet seria uma plataforma para desenvolvimento de aplicações baseadas em serviços;
2. Desenvolvimento de um Timeline Web Browser, um browser que grava as ações do usuário realizadas na internet e as disponibiliza através de web services, permitindo a criação de novos serviços.

1.2.Trabalhos Existentes

A pesquisa de referências bibliográficas mostrou que existem diversos artigos que tratam dos conceitos de Web 2.0.

Existem também livros e artigos que apresentam as tecnologias de *Web Services* e de *Service Oriented Architecture.* Porém, muitos destes documentos apresentam estes conceitos atrelados a tecnologias específicas, como Java ou .NET.

Este trabalho não possui precedente, pois tem como objetivo unir todos estes conceitos na criação de uma arquitetura padrão independente de produtos específicos.

1.3. Visão Geral

Este documento está dividido nos seguintes capítulos:

- **Introdução:** apresenta o contexto no qual o trabalho se insere, bem como seus objetivos.
- **Web 2.0:** apresenta uma visão geral do termo Web 2.0, sua história e trata mais a fundo a definição tratada neste trabalho – a web como plataforma de software.
- **Tecnologias da plataforma Web:** apresenta as principais soluções e tecnologias usadas para se construir a plataforma web.
- **Arquitetura de uma aplicação Web 2.0:** apresenta uma arquitetura independente de soluções proprietárias para construção de aplicações de acordo com o conceito de web como plataforma de software.
- **Timeline Web Browser:** descreve um Timeline Web Browser e sua arquitetura.
- **Conclusões e considerações finais:** conclui o trabalho e apresenta pontos de continuidade.

2. Web 2.0

"The world is changing."
TreeBeard (The Lord of the Rings: The Two Towers)

O termo Web 2.0 se refere a uma segunda fase de desenvolvimento da web, no que diz respeito a sua arquitetura, suas aplicações e principalmente, suas formas de uso. O termo está associado a vários significados, entre eles:

- A transição de sites estáticos para gerenciadores de conteúdo e funcionalidades, transformando a web em plataforma computacional;
- Um fenômeno social relacionado à criação e distribuição de conteúdo na web, de forma aberta e descentralizada;
- Uma mudança no valor econômico da web;
- Um termo de marketing, para diferenciar os novos negócios da web daquelas da bolha da internet.

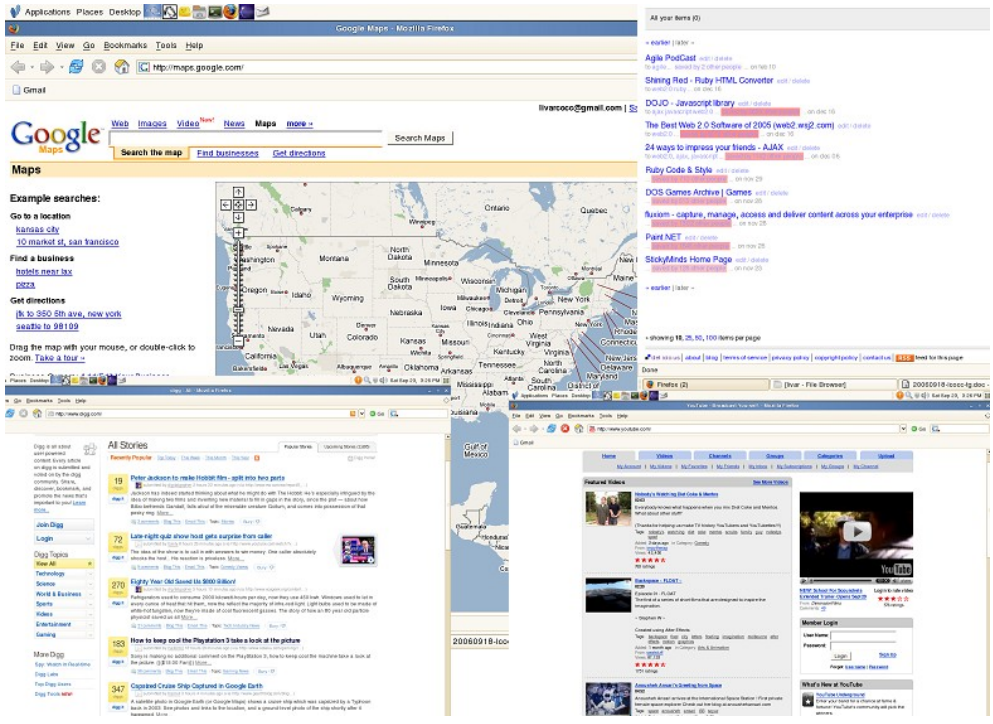

Figura 1: Google Maps, del.icio.us, digg, Flickr, Youtube

No entanto, ainda não se chegou a um consenso em relação ao significa do termo. Talvez, uma forma mais simples de entender o significado de Web 2.0 seja

através da associação do termo com produtos baseados nos seus princípios. Alguns desses produtos mais conhecidos são Google Maps , del.icio.us , digg , Flickr e Youtube .

2.1.Visão Geral

A versão anterior da web (Web 1.0) consistia basicamente de páginas HTML estáticas que eram raramente atualizadas, se tanto. O sucesso da "Era .com" dependia de uma web mais dinâmica, onde sistemas de gerenciamento de conteúdo (CMS) geravam páginas HTML dinâmicas a partir de um banco de dados que podia ser atualizado mais facilmente. Em ambos os casos, a questão visual era considerada intrínseca ao uso da web, tornando estética visual um fator de extrema importância. A Web 2.0 transforma a aparência das aplicações web, deixando-as com cara de aplicações desktop e elevando o conceito de páginas dinâmicas. Alguns exemplos de produtos, onde isso acontece de forma mais explicita, são ThinkFree , Writely , EyeOS e NetVibes .

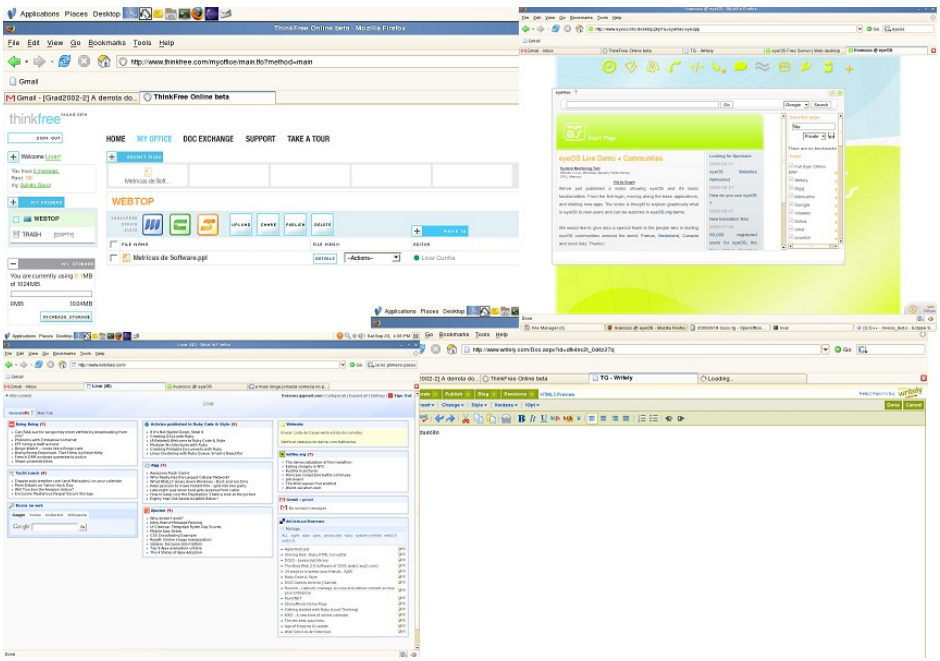

Figura 2: ThinkFree, Writely, EyeOS, Netvibes

A web é, cada vez mais, baseada em interação e em redes sociais rudimentares. De certa forma, sites Web 2.0 são como portais dependentes de usuários. A web está cada vez menos sob o controle de especialistas e se aproximando mais do seu conceito original de meio de comunicação democrático e pessoal.

Com uma alusão aos números de versões, normalmente usados para indicar novas versões de software, o termo "Web 2.0" indica uma forma aprimorada da *World Wide Web.* O termo foi popularizado por uma conferência patrocinada pela O'Reilly Media e pela MediaLive depois que Dale Dougherty, mencionou o termo em uma sessão de *brainstorming*. Dougherty afirmou que a web estava passando por uma revolução.

2.2.Um pouco de história

Na abertura da primeira conferência sobre Web 2.0, O'Reilly e Battelle resumiram as características principais das aplicações 2.0:

- A web como uma plataforma, onde a web se coloca como uma provedora de serviços que podem ser usados para criar novas aplicações;
- Dados como a força principal;
- Efeitos de rede criados por uma "Arquitetura de participação" ;
- Inovação na montagem de sistemas e sites. Onde a junção (mashup) de dois serviços comuns, gera um novo, completamente inovador;
- Modelo de programação leve, a partir do uso de web services e baixo acoplamento;
- Fim do ciclo de release de software, onde software passa a ser um serviço, ao invés de um produto, exigindo das companhias uma manutenção diária de seus serviços;
- Software acima do nível de um único dispositivo, ou seja, não está mais limitado a uma plataforma PC.

Assim, uma forma de uso inicial do termo Web 2.0 se referia a "Web Semântica", e, de certa forma, os dois conceitos se complementam. A combinação de sistemas de redes sociais, com o desenvolvimento de folksonomias[1] baseadas em tags e disponibilizadas através de *Blogs* e *Wikis* cria uma base natural para um ambiente semântico. Apesar das tecnologias e serviços da Web 2.0 serem menos poderosos que

[1] Folksonomia é um neologismo para a prática de categorização colaborativa usando palavras de livre escolha.

uma internet onde as máquinas podem entender e extrair significados, a Web 2.0 representa um passo nessa direção.

2.3.Tecnologias Web 2.0

A infra-estrutura da Web 2.0 é complexa e está em evolução. Ela inclui softwares baseados em servidores, protocolos de comunicação, browsers baseados em padrões e várias aplicações cliente. Estas abordagens diferentes e complementares provêem a Web 2.0 com capacidades de criação, disseminação e armazenamento de informações além da imaginada inicialmente para páginas web.

Exemplos de tecnologias comumente usadas em sites Web 2.0 são:

- Ajax
- XUL e SVG
- RSS
- SOA e Web Services

2.3.1.Plataforma Web

Web Platform (Plataforma Web) foi o nome da primeira conferência promovida pela O'Reilly e pela MediaLive tratando do tema Web 2.0. Durante a conferência eles perceberam que Web 2.0 ia muito além deste conceito inicial, mas isto não tornou este aspecto menos importante.

O conceito de plataforma web se baseia na idéia de aplicações que são desenvolvidas usando serviços disponibilizados na internet. Assim, a web funciona de forma muito semelhante a um sistema operacional, como Windows, que fornece funções (serviços) para que terceiros possam desenvolver novas aplicações e funcionalidades.

A web fornece seus serviços através de APIs (*Application Programming Interface*), que são conjuntos de funções usadas pelos programadores. Exemplos de APIs são as disponibilizadas na MFC (*Microsoft Foundation Classes)*, usadas para desenvolver aplicações windows; a API de java.io, que fornece funções para manipulação de arquivos em Java; ou ainda, a API do *Google Search*, disponibilizada

através da web para que outros desenvolvedores possam fazer uso da engine de busca do Google em seus próprios sites, desenvolvendo novos e inovadores serviços.

As aplicações web que disponibilizam ou utilizam APIs na web implementam a arquitetura SOA (*Service Oriented Architecture)* através da tecnologia de web services. Web services usam XML para descrever as funções da API e para realizar o transporte de informações do cliente para o servidor, e vice-versa, de forma independente de plataforma e de linguagem de programação, garantindo a heterogeneidade da web.

3. Tecnologias da Plataforma Web

"Live in fragments no more. Only connect…"
E.M. Forster

O desenvolvimento na plataforma web exige que serviços possam ser desenvolvidos e usados de forma independente de plataforma e de linguagem. Essa necessidade se apresenta pelo fato da web ser um ambiente extremamente heterogêneo. As tecnologias descritas abaixo atendem perfeitamente a este requisito e são as mais largamente usadas. Uma indicação disso é um relatório da Gartner que diz que "Em 2008, SOA será uma prática prevalecente de engenharia de software, acabando com um domínio de 40 anos de arquiteturas de software monolíticas" e ainda que "Durante 2008, SOA e web services serão implementados juntos em mais de 75% dos projetos que usarem SOA ou web services".

Estas tecnologias de forma alguma cobrem todas as tecnologias relacionadas à Web 2.0, mas são as tecnologias básicas mais usadas para o desenvolvimento de aplicações que se encaixam no conceito de plataforma web.

3.1.Arquitetura SOA

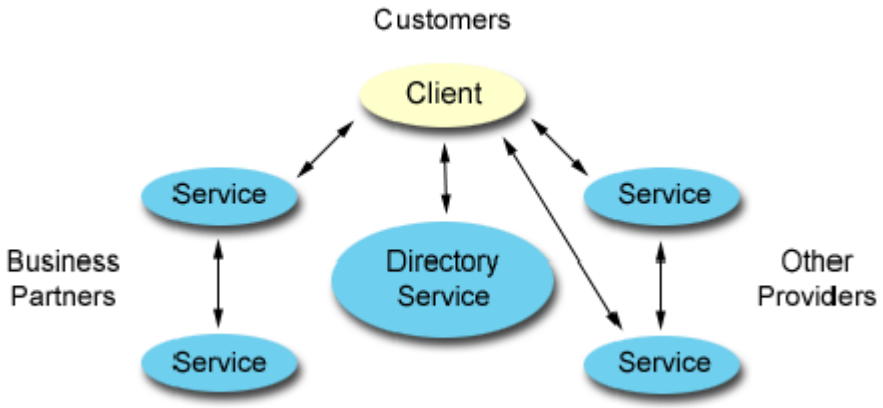

Figura 3: Arquitetura SOA

Uma arquitetura SOA (Service-Oriented Architecture) é uma abordagem na qual as aplicações fazem uso de serviços disponíveis em uma rede, como a World Wide Web. Implementar uma arquitetura SOA envolve o desenvolvimento de

aplicações que usam serviços ou que são disponibilizadas como serviços, de forma que outras aplicações posam usar estes serviços e vice-versa.

Um serviço pode prover uma única funcionalidade simples, como realizar a conversão de valor de uma moeda para outra, até pode provar um conjunto de funções complexas, como lidar com as várias operações em um sistema de reservas de passagens aéreas.

Uma maneira de olhar para a SOA é como uma abordagem para conectar diferentes aplicações (disponibilizadas como serviços) de forma que elas possam se comunicar umas com as outras. Em outras palavras, SOA é uma forma de compartilhar funções (serviços) de uma maneira abranjente e flexível. E esta é exatamente a sua principal característica para a Web 2.0.

O conceito de SOA não é novo. Service-*Oriented Architectures* têm sido usadas por anos. O que diferencia SOA de outros modelos de arquitetura é o baixo acoplamento. Esse baixo acoplamento significa que o cliente de um serviço é, essencialmente, independente do serviço. A forma como o cliente (que pode ser até um outro serviço) se comunica com o serviço independe da maneira como o serviço foi implementado. Isso significa que o cliente não precisa saber a linguagem na qual o serviço foi implementado ou em qual plataforma o serviço está rodando, característica essencial para a plataforma web. Na SOA o cliente se comunica com o serviço através de uma interface bem definida, e deixa que o serviço em si realize as operações necessárias. Se a implementação de um serviço mudar, o cliente continua se comunicando com o serviço da mesma maneira, desde que a interface continue a mesma.

No entanto, o que é relativamente novo em SOA são os web services baseados em SOA. Um web service é um serviço que se comunica com um cliente através de um conjunto de protocolos e tecnologias padrões. Estes padrões de web services são implementados por todos os grandes vendedores de software, tornando possível para clientes e serviços se comunicarem de forma consistente dentro de uma grande gama de plataformas e sistemas operacionais. Estas adoções universais dos web services os torna a abordagem mais comum para implementar uma SOA.

3.2. Web Services

Web Services representam uma nova geração de tecnologia de desenvolvimento. Com ela é possível criar aplicações modulares e independentes que são facilmente distribuídas em qualquer estrutura de redes TCP/IP, pois este foi um dos principais fundamentos de sua implementação.

Um grande ponto positivo desta tecnologia é que a criação de servidores e clientes independe da linguagem de programação e do sistema operacional em que são implementados. Isto torna este tipo de tecnologia ideal para o ambiente heterogêneo da web.

A abordagem de web services é baseada em padrões que são amplamente aceitos e usados. Esta aceitação abranjente torna possível que clientes e serviços se comuniquem e se entendam mesmo em uma variedade de plataformas e independente de linguagens. Os principais padrões e protocolos usados por web services são: XML, SAOP, WSDL e UDDI e ebXML, descritos a seguir.

3.2.1. XML

eXtensible Markup Language (XML) se tornou o padrão para troca de dados na Web. Como o próprio nome indica, XML é uma linguagem de marcação, como HTML. Ela faz uso de tags para "marcar" a o conteúdo de um documento. Uma tag XML identifica a informação dentro de um documento, e também identifica (ou marca) a estrutura da informação. Por exemplo, a estrutura XML a seguir identifica informações a respeito de uma estante de livros. Os marcadores XML também descrevem a estrutura da estante. Nesta estrutura, um elemento <*estante*>, possui um subelemento <*livro*>, que por sua vez possui três outros subelementos, <*titulo*>, <*autor*> e <*preco*>.

```
<estante>
    <livro>
        <titulo>The SOA within reach</titulo>
        <autor>Dion Hinchcliffe</autor>
        <preco>50.00</preco>
    </livro>
</estante>
```

Listagem 1: Exemplo de XML

Apesar das tags acima possuírem, aparentemente, um significado inerente para a informação que eles identificam, elas não possuem. Informações marcadas em XML possuem significa apenas se as pessoas associarem significados as tags XML. Apenas se as pessoas concordarem com os significados das tags e usarem as tags de forma consistente XML poderá prover uma forma consistente de troca de informações. Suas aplicações podem trocar documentos XML entre si, e processar a informação nestes documentos, confiando nos significados que foram estabelecidos para as tags.

Documentos XML possuem uma estrutura bem formada, ou seja, toda tag XML deve possuir uma tag de fechamento (<*estante*>...</*estante*>) e qualquer tag que comece dentro de uma outra tem que fechar antes do fim da outra. Tipicamente, um documento XML é associado a um esquema (*schema*) que especifica suas "regras gramaticais". Este documento especifica que tags são permitidas dentro do documento XML, a estrutura destas tags, além de outras regras.

Como documentos XML válidos devem ser bem formados e estar em conformidade com o esquema a que estão associados, eles se tornam relativamente fáceis de processar. Como resultado, XML é o padrão geralmente adotado para troca de dados em web services.

3.2.2.SOAP

Apesar de XML ser um meio efetivo para troca de informações, não é o suficiente para a troca de dados na web. Por exemplo, ainda é necessário definir um protocolo para a formatação de um documento XML, de forma que o receptor entenda o que é a parte principal da mensagem, "payload", e que partes contêm instruções e conteúdo adicionais.

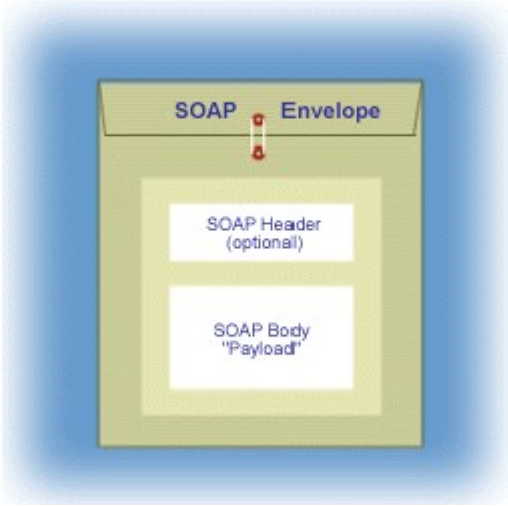

Figura 4: Uma mensagem SOAP

Este protocolo é o *Simple Object Access Protocol* (SOAP). SOAP é um protocolo baseado em XML para troca de informações em um ambiente distribuído, como a web. SOAP estabelece um formato de mensagem comum para a troca de dados entre cliente e serviço.

O elemento básico para uma transmissão usando SOAP é a mensagem (*SOAP Message*), que é formada por um elemento obrigatório, o envelope (*SOAP Envelope*), por um cabeçalho opcional (*SOAP Header*), e mais um elemento obrigatório, o corpo da mensagem.

O envelope é formado por dois elementos, um *XML namespace* e um *encoding style*. O XML namespace especifica os nomes que podem ser usados dentro da mensagem e evitam conflitos de nomes. O *encoding style* define os tipos de dados reconhecidos pela mensagem.

Se a mensagem possuir um cabeçalho, ele será usado para estender a mensagem de forma modular. É importante entender que uma mensagem viaja de um cliente para um servidor, e que no caminho ela passa por um conjunto indeterminado de nós. Cada nó é uma aplicação que pode receber e encaminhar mensagens SOAP. E um nó intermediário pode prover serviços adicionais, como realizar operacionais relacionadas à segurança. O cabeçalho pode ser usado para indicar que é necessário realizar algum processamento extra em um dos nós intermediários. Este processamento seria independente do processamento feito no nó final.

O corpo contém a maior parte de uma mensagem, ou seja, a parte destinada ao nó de destino. O exemplo abaixo mostra uma mensagem SOAP com o objetivo de

obter o preço de um livro. Os elementos <SOAP-ENV: Envelope>, <SOAP-ENV: Header> e <SOAP-ENV: Body> marcam respectivamente o envelope, o cabeçalho e o corpo da mensagem. O elemento <SOAP-ENV: mustUnderstand=”1”> dentro do cabeçalho indica que o receptor da mensagem deve processá-la obrigatoriamente.

```
<SOAP-ENV: Envelope
    xmlns:SOAP-ENV=
        http://schemas.xmlsoap.org/soap/envelope/
    SOAP-ENV:encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/">

    <SOAP-ENV:Header>
        <t:Transaction xmlns:t="some-URI">
            <SOAP-ENV:mustUnderstand="1">
        </t:Transaction>
    </SOAP-ENV:Header>

    <SOAP-ENV:Body>
        <m:GetBookPrice xmlns:m="some-URI">
            <title>My Life and Times</title>
            <author>Felix Harrison</author>
        </m: GetBookPrice>
    </SOAP-ENV:Body>

</SOAP-Envelope>
```

Listagem 2: Exemplo de Mensagem SOAP

Mensagens SOAP são independentes de plataforma e sistema operacional, e podem ser transportadas usando vários protocolos, por exemplo, HTTP ou SMTP.

3.2.3.WSDL

Como um cliente sabe que formato usar quando fizer uma requisição de um serviço? Como o cliente e o serviço sabem o que uma requisição significa? As respostas para estas perguntas estão em um documento XML chamado de documento WSDL. Este documento contém a descrição da API de um web service . WSDL define um esquema XML para descrição de um web service. Para descobrir a especificação de um web service, um cliente precisa encontrar o documento WSDL deste serviço. A forma mais típica de localizar este documento é através de um ponteiro para este documento que pode ser encontrado no registro do web service, que pode estar em um registro UDDI ou em um registro/repositório ebXML.

Por exemplo, alguém registra um serviço. O registro inclui um ponteiro para um arquivo WSDL que possui o documento WSDL para este serviço. Um cliente

procura o registro e encontra o serviço. Um programador usa as informações no documento WSDL para conhecer a interface do serviço e realizar as chamadas necessárias às funções desta interface.

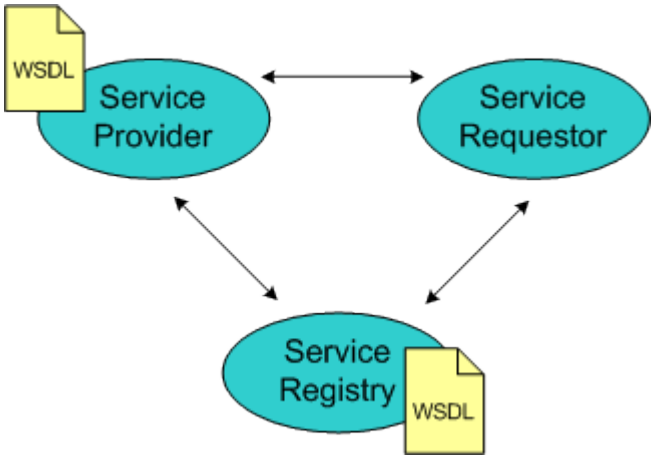

Figura 5: Comunicação cliente-serviço usando WSDL

Um documento WSDL descreve um web service como uma coleção de itens abstratos chamadas de *ports* ou *endpoints*. Ele também descreve, de forma abstrata, as ações realizadas por um web service e as informações transmitidas para estas ações. Ações são representadas por operações e dados são representados por mensagens. Uma coleção de operações é chamada de *port type*.

O que transforma uma descrição WSDL de abstrata em concreta é um "binding". Um binding especifica um protocolo de rede e um formato de mensagem para um port type em particular. Um port é definido associando um endereço de rede com um binding.

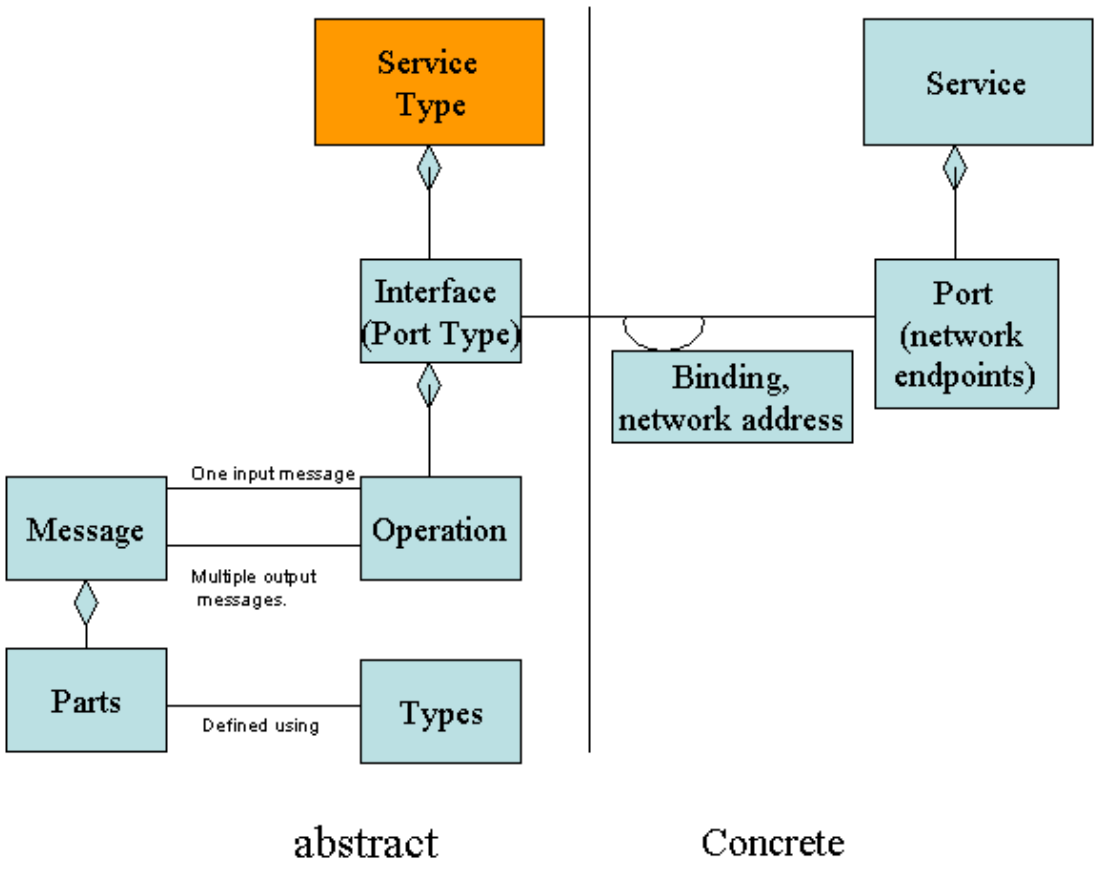

Figura 6: Diagrama de Objetos WSDL

Se um cliente localizar um documento WSDL e encontrar o binding e o endereço de rede para cada porta, ele pode chamar as operações do serviço de acordo com o protocolo e o formato de mensagens especificado.

O Apêndice A possui um exemplo de um documento WSDL que especifica um serviço de busca de livros online. No exemplo, uma operação é especificada para o serviço:

```
<operation name="getBooks" ...
```

Listagem 3: Especificação de uma operação para um serviço

A mensagem de entrada para a operação, BookSearchInput possui uma string chamada isbn:

```
<complexType>
    <all>
        <element name="isbn" type="string"/>
    </all>
```

Listagem 4: Especificação de um dado de entrada para operação WSDL

A mensagem de saída da operação, BookSearchOutput retorna uma string chamada title:

```
<complexType>
    <all>
        <element name="title" type="string"/>
    </all>
```

Listagem 5: Especificação de um dado de saída para operação WSDL

3.2.4.UDDI

Como foi mencionado anteriormente, uma arquitetura SOA pode possuir também um serviço que provê um diretório ou registro de serviços. Mas como um serviço pode ser descrito em um registro de forma que um outro registro possa facilmente localizá-lo? O *Universal Description, Discovery and Integration* (UDDI) descreve como publicar e recuperar informações sobre serviços em um registro baseado em UDDI.

O registro UDDI pode ser visto como um conjunto de "Páginas Amarelas" de web services. Como as Páginas Amarelas de números de telefone, o registro UDDI provê informações em relação a um serviço, como o nome do serviço, uma breve descrição do que ele faz, um endereço onde o serviço pode ser acessado, e uma descrição da interface para acessar este serviço.

4. Arquitetura de uma aplicação Web 2.0

"Imagination is more important than knowledge"
Isaac Newton

A arquitetura proposta neste capítulo foi projetada com o objetivo de permitir o desenvolvimento de aplicações baseadas no conceito de web como plataforma de software. Por isso, essas aplicações devem ser capazes de utilizar serviços disponíveis na web, bem como disponibilizar serviços próprios também na web.

Além desta característica, a arquitetura também tem como objetivo ser genérica o bastante para poder ser usada no desenvolvimento de outras aplicações para a plataforma web.

Dois padrões arquiteturais serão usados para definir a arquitetura geral. O padrão em camadas (Layers pattern) , que ajuda a separar os vários conceitos presentes no tipo de aplicação que a arquitetura visa atender. E o padrão de N-Camadas (N-Tier architecture pattern), uma derivação do padrão em camadas, com uma aplicação mais específica.

4.1.Introdução ao padrão em camadas

O padrão em camadas é básico por natureza, mas extremamente importante na prática. O conceito essencial do padrão é que as aplicações devem ser projetadas divididas em camadas, com cada camada representando um conjunto de conceitos relacionados. Cada camada possui classes e componentes relacionados, e possui uma interface estruturada e um mecanismo de comunicação para acessar as capacidades de uma outra camada.

As camadas tipicamente se empilham umas em cima das outras, construindo as funcionalidades gerais da aplicação, como mostra a Figura 7. É comum que uma camada só acesse a camada imediatamente inferior a ela, apesar de que, comumente, as camadas tenham acesso a múltiplas camadas na prática.

A parte mais difícil em trabalhar em camadas é determinar quantas camadas a aplicação será separada e como realizar a separação entre elas. A separação em

camadas deve fazer sentido de um ponto de vista técnico e também de um ponto de vista de estrutura organizacional. É necessário manter em mente que o aspecto técnico prevalecerá sobre o aspecto organizacional, portanto é necessário ter cuidado ao considerar o mérito técnico como principio organizacional primário. Camadas típicas são: a camada de apresentação, camada de negócios, camada de dados e camada cliente. Por exemplo, as preocupações da camada de apresentação com o browser são completamente diferentes das preocupações da camada de dados com o banco de dados.

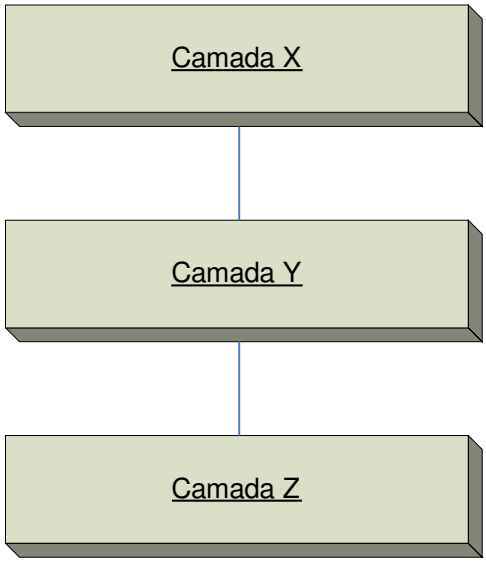

Figura 7: Padrão arquitetural em camadas

É possível separar as camadas de várias formas diferentes. Também é possível implementar a arquitetura em camadas e ainda assim rodar toda a aplicação em um único processo físico. Também é possível separar as camadas em processos em um único computador, ou ainda, separar as camadas em processos em diferentes computadores, essencial para a implementação nos conceitos da plataforma web.

No entanto, apenas dizer que a arquitetura de uma aplicação é em camadas não é o suficiente para os desenvolvedores e projetistas. É necessário ir além e explicar os conteúdos de cada camada na arquitetura .

4.2.Introdução ao padrão arquitetural de N-Camadas

O padrão arquitetural de N-Camadas é uma generalização do padrão arquitetural de três camadas. O padrão arquitetural de três camadas é formado por uma camada de apresentação, uma camada de lógica do negócio e uma camada de persistência de dados. No entanto, uma arquitetura de três camadas não possui espaço para uma camada de comunicação, própria para os web services, necessários a disponibilização de serviços para uso de terceiros.

Uma descrição das camadas, assim como uma descrição de como essas camadas se comunicam, tipicamente fazem parte da descrição de uma arquitetura de N-Camadas. A Figura 8 mostra um exemplo que possui uma camada cliente, uma camada de apresentação, uma camada de lógica do negócio e uma camada de persistência de dados. A comunicação entre a camada cliente e a camada de apresentação pode acontecer usando *Hypertext Transfer Protocol* (HTTP), a comunicação entre a camada de apresentação, a camada de negócios pode acontecer usando *Remote Method Invocation* (RMI) e a comunicação entre a camada de negócios e a camada de dados pode acontecer usando JDBC (É importante ressaltar que as tecnologias citadas acima são apenas exemplo).

No entanto, uma aplicação que funcione de acordo com os conceitos de plataforma web, precisa ter seus serviços acessados a partir de qualquer outra aplicação na web, ou seja, ela possui múltiplos pontos de acesso ao sistema. Os usuários podem usar o sistema a partir de uma interface gráfica, e programas podem acessar o sistema a partir de APIs. O desenvolvedor do sistema, e dos serviços, não sabe todas as formas que os clientes da aplicação vão usar seus serviços; eles podem desenvolver sua própria lógica, ou sua própria camada de apresentação baseadas nesses serviços.

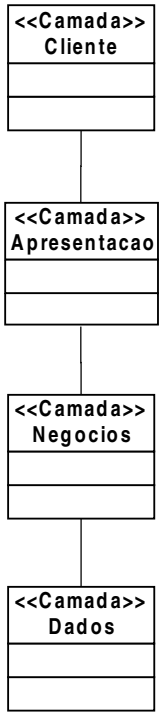

Figura 8: Padrão arquitetural de N-Camadas

4.3.Aplicações baseadas em serviços

A arquitetura base para aplicações baseadas na plataforma web são um desdobramento do padrão arquitetural de N-Camadas. A Figura 9 mostra a estrutura da aplicação e da plataforma em alto nível. A arquitetura segue o padrão arquitetural de N-Camadas, que é derivado do padrão arquitetural em camadas, e é comum em várias aplicações baseadas na plataforma web.

A estrutura da arquitetura proposta consiste nas seguintes camadas:

- Camada Cliente: Esta camada contém os diversos dispositivos e mecanismos que os clientes da aplicação usarão para acessar os serviços sistema. Suas principais formas de acesso são através de um browser e através das funções da API disponibilizadas utilizando-se web services, apesar de outros mecanismos também poderem ser usados, como uma GUI. Os principais protocolos de suporte usados são *Extensible HTML* (XHTML), em cima de HTML, para acessos através do browser, e SOAP, para acesso por aplicações de terceiros. A aplicação deve ser flexível o bastante para se adaptar a outras formas de acesso.

- Camada Web: Esta camada possui duas responsabilidades: aceitar as requisições feitas a partir de browsers clientes e aceitar requisições de outras camadas de negócios feitas a partir da API disponibilizada pela aplicação. Ambos os tipos de requisição são passados para a camada de negócios para processamento. É nesta camada que está presente também a implementação dos web services.
- Camada de Negócios: Esta camada possui a lógica da aplicação necessária para atender aos requisitos do sistema. No entanto, existem casos em que a lógica do negócio existe fora da aplicação. Neste caso, esta camada possui a lógica necessária para integrar a lógica externa a aplicação.
- Camada de Dados: Esta camada, presente na maioria das aplicações, é responsável pela persistência dos dados, através de um banco de dados, de arquivos ou semelhantes.

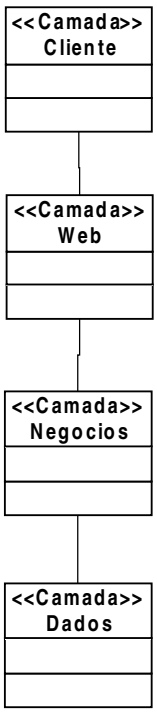

Figura 9: Arquitetura das aplicações da plataforma web

Apesar da importância das camadas cliente e de dados, elas não sofrem grandes modificações devido ao conceito de plataforma web. No entanto, as camadas web e de negócios são fortemente modificadas pela inserção dos web services, na

verdade, a camada web é inserida praticamente apenas para acomodar essa modificação, por isso essa camada será tratada com um pouco mais de detalhe.

4.3.1.Camada Web

Esta camada é responsável pela comunicação entre as requisições das aplicações clientes e a lógica de negócios da aplicação. A interface que será acessada por terceiros e disponibilizada através de web services é implementada nesta camada.

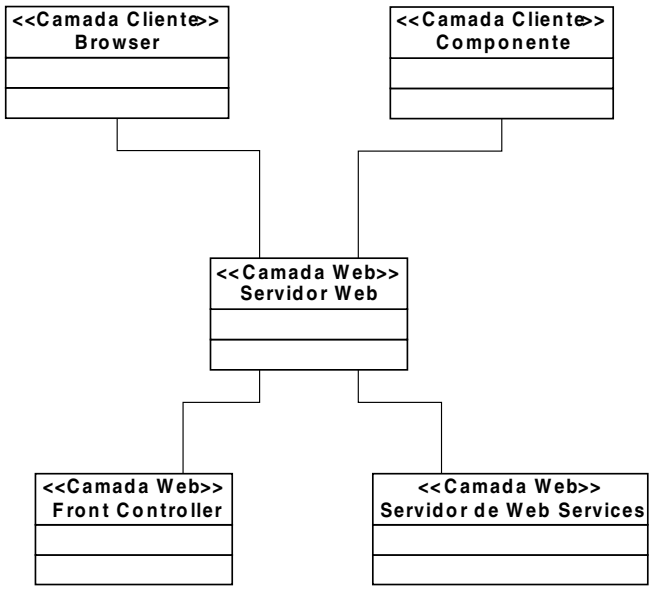

Figura 10: Camada Web

Esta camada pode atender a duas funções especificas: o acesso a esta camada através de um browser, neste caso a camada processará as requisições e respostas para o browser; e o acessa a partir de uma API, como ilustra a Figura 10. O tratamento das chamadas a partir da API é feito pela interface dos web services. O servidor encaminha as requisições para um servidor de web services, como o Apache Axis . Requisições feitas por uma camada de apresentação, a partir de um browser, por exemplo, são redirecionadas para um *Front Controller*, um controller que cuida de todas as requisições feitas a um site . O Front Controller facilita a implementação do padrão *Model-View-Controller* na camada de apresentação, o mais comumente usado em aplicações web hoje em dia.

5. Timeline Web Browser

"Anything that works is better than anything that doesn't"
Autor desconhecido

Timeline Web Browser é uma aplicação baseada no conceito de plataforma web (no caso ela disponibiliza seus serviços para que terceiros possam desenvolver suas próprias aplicações) construída de acordo com a arquitetura estabelecida no capítulo anterior.

O Timeline Web Browser é uma aplicação que guarda um histórico de todas as ações realizadas pelo usuário no browser e disponibiliza essas informações na web, como o del.icio.us faz com os favoritos de seus usuários. Além de gravar e disponibilizar este histórico, a aplicação também permite executar este histórico, de forma semelhante a um vídeo e inserir comentários neste histórico. Também é possível realizar busca nestas informações usando como critério o tempo (data e hora), permitindo que uma pessoa possa saber exatamente o que estava fazendo em uma data específica, daí o nome de Timeline Web Browser.

As funcionalidades de registrar as ações realizadas no browser, inserir comentários e executar buscas nestas informações devem ser disponibilizadas para terceiros através de um API usando-se web service.

Em resumo, as funcionalidades do Timeline Web Browser são as seguintes:

- Gravar as ações realizadas pelo usuário no browser;
- Realizar um *playback* dessas ações
- Inserir comentários, ou outras ações, no timeline;
- Realizar buscas nos dados usando como critérios data e hora[2];
- Disponibilizar as ações acima, com exceção a de playback, em forma de API na web.

[2] Está funcionalidade não será implementada nesta versão do Timeline Web Browser por restrições de tempo.

5.1.Arquitetura do Timeline Web Browser

A aplicação foi desenvolvida na linguagem de programação Java. Utiliza o servidor de aplicação Apache Tomcat e o servidor de web services Apache Axis e como servidor de banco de dados o MySQL.

O Timeline Web Browser está organizado nas seguintes camadas: Cliente, Web, Negócio e Dados como descrito no capítulo anterior. Estas camadas serão explicadas em detalhes a seguir, descrevendo as tecnologias e padrões usados em cada camada. A seguir também será descrita a estrutura de pacotes utilizada.

5.1.1.Estrutura de pacotes

O Timeline Web Browser está estrutura em quatro pacotes:

- timelinewebbrowser: Este pacote, além de ser um pacote comum a todos os outros pacotes, possui as classes da camada Cliente da aplicação. No caso, a classe *TimelineWebBrowser.*
- webservice: Este pacote possui a fachada da aplicação.
- composite: Este pacote possui as classes de negócio da aplicação - as classes que representam os itens do histórico.
- util: Este pacote possui classes utilitárias que poderão ser usadas por quaisquer outros pacotes.

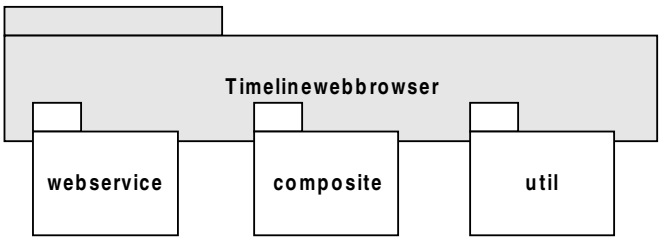

Figura 11: Estrutura de pacotes

5.1.2.Camada Cliente

Esta camada é responsável pela interface com o usuário. Ela repassa as ações do usuário para a API presente na camada Web.

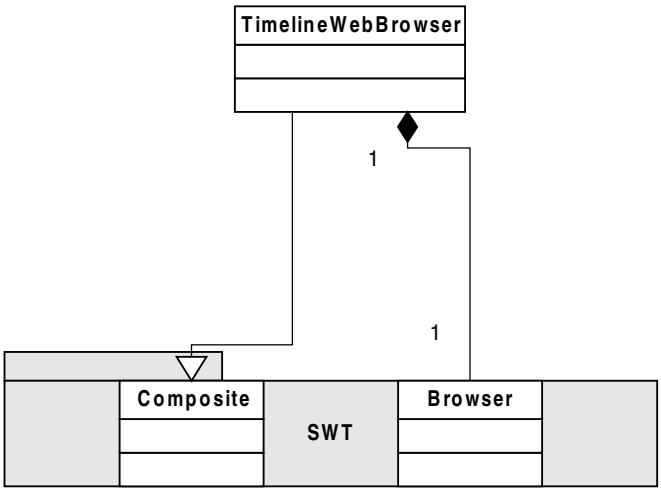

Figura 12: Camada Cliente do Timeline Web Browser

Esta camada foi desenvolvida utilizando-se a bliblioteca SWT e está organizada como descrito na Figura 12: a classe *timelinewebbrowser.TimelineWebBrowser* extende a classe *org.eclipse.swt.widgets.Composite* e possui como componente uma *org.eclipse.swt.browser.Browser*. A classe TimelineWebBrowser é responsável por tratar as ações realizadas na classe Browser e em seguida repassar estas ações para a camada Web. Esta camada por sua vez encaminha estas ações para a camada de negócio, que as trata e então envia uma resposta de volta para a camada Cliente, também através da camada Web.

5.1.3.Camada Web

Esta camada implementa a infra-estrutura de web services responsável por receber as requisições da camada Cliente e repassar estas requisições para a camada de Negócios, bem como enviar as respostas de volta para a camada Cliente.

Estão presentes nesta camada basicamente o servidor de aplicações Apache Tomcat, o servidor de web services Apache Axis e a classe *timelinewebbrowser.webservice.TimelineWebBrowserFacade*, que implementa o padrão *Facade* e funciona como porta d entrada para as funções disponíveis no web service. A comunicação cliente-servidor ocorre de acordo com a Figura 13: as chamadas do cliente são codificadas em SOAP e transmitidas por HTTP (eventuais firewalls não bloqueiam a requisição); quando o serviço recebe uma requisição

acontece o inverso, as chamadas são decodificadas e executadas, e a resposta é codificada novamente para ser enviada para o cliente.

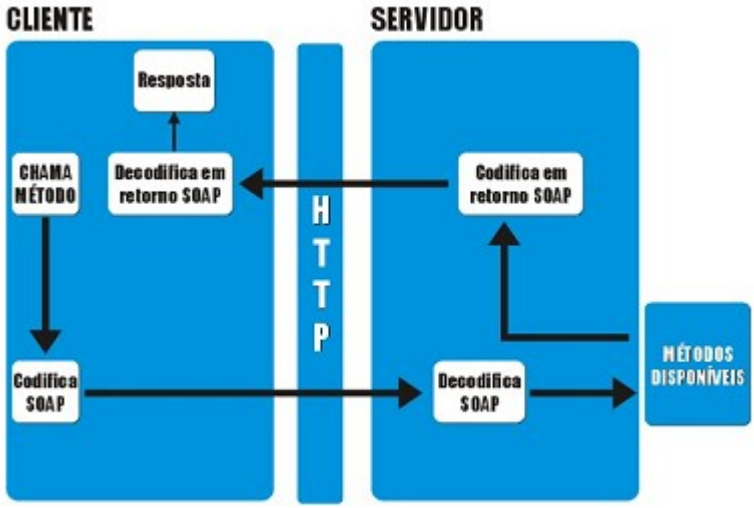

Figura 13: Comunicação Cliente-Servidor

As aplicações clientes descobrem os métodos disponíveis no web service através de arquivos WSDL, já descritos anteriormente. Um exemplo de uso de web services no Timeline Web Browser é mostrado na Listagem 6. Este código constrói uma chamada para método de um web service que está no localhost, chamado *addHistory*, e então invoca este método passando como parâmetros o tipo de histórico (GO) e a url do site.

Parte do arquivo WSDL para a função de web service acima está descrito na Listagem 7. Este código acima é muito simples. Ele informa um nome para o web service (<service namae>), o nome da classe (“class name”), que é definido no primeiro parâmetro, e os métodos disponibilizados como web services (“allowedMethods”), como descrito no segundo parâmetro (o asterisco indica que todos os métodos públicos serão disponibilizados).

Isto cobre todos os elementos da camada Web: um servidor de aplicações, junto com um servidor de web services, e, por final, os arquivos WSDL de descrição dos web services.

```
…
private static String ENDPOINT_ADD_HISTORY =
  “http://localhost:5049/axis/services/addHistory”;
public class TimelineWebBrowser {
  private Button createGoButton(Composite controls) {
```

```
    if (goButton == null) {
      goButton = new Button(controls, SWT.PUSH);
      goButton.setText(TimelineWebBrowser.GO_BUTTON);
      goButton.addSelectionListener(new SelectionAdapter(){
        public void widgetSelected(SelectionEvent e) {
          browser.setUrl(url.getText());
          Service service = new Service();
          call call = (Call)service.createCall();
          call.setTargetEndpointAddress(
            new java.net.URL(ENDPOINT_ADD_HISTORY));
          call.setOperationName(
            new QName("http://localhost/", "addHistory"));

            call.invoke (new Object[] {"GO", url.getText()});
        }
      });
    }
    return goButton;
  }
}
```

Listagem 6: Invacação de um web service

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <service name="TimelineWebBrowser" provider="java:RPC">
       <parameter name="className" value="TimelineWebBrowserFacade"/>
       <parameter name="allowedMethods" value="*"/>
    </service>
</deployment>
```

Listagem 7: Arquivo WSDL de um serviço do Timeline Web Browser

5.1.4.Camada de Negócios

Esta camada é responsável por implementar as regras de negócio do Timeline Web Browser. Ela implementa o playback de objetos, lendo as informações da camada de Dados e repassando para a camada Cliente, por intermédio da camada Web. Ela também é responsável por inserir comentários em um elemento do histórico.

5.1.4.1.Playback

A funcionalidade de playback é construída a partir do padrão de projetos *Composite* . Este padrão permite a composição de objetos na forma de parte–todo, como mostra a Figura 14. O todo é chamado para executar o playback, e os objetos não-folha controlam a ordem de execução de suas partes de forma recursiva. Os objetos folha, por sua vez são responsáveis por executar o playback propriamente.

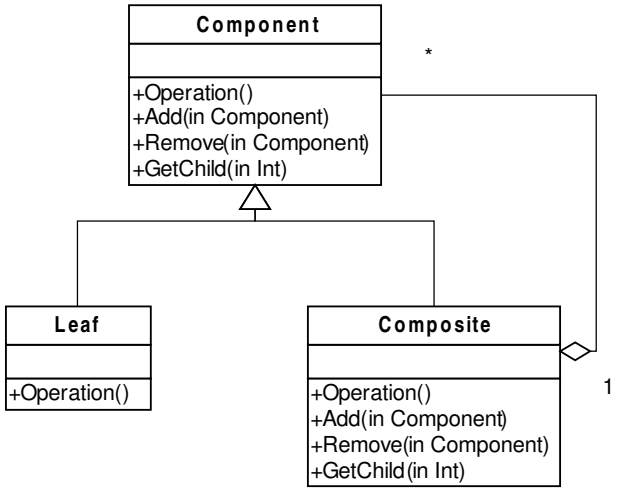

Figura 14: Padrão de Projeto Composite

Este tipo de estrutura também permite a extensão do Timeline Web Browser de forma a suportar novos objetos de playback, bastanto estender um dos itens da hierarquia. No caso Timline Web Browser, a classe *timelinewebbrowser.composite.History* representa a superclasse do padrão Composite. O elemento composto é implementado pela classe *timelinewebbrowser.composite.CompositeHistory*. Está classe será composta por vários objetos *History* e sua função de playback chama a função de playback de todos os objetos que a compõe. Alguns dos elementos folha são *timelinewebbrowser.composite.TypedURL* e *timelinewebbrowser.composite.Link*. A Figura 15 mostra a estrutura geral do padrão aplicado ao Timeline Web Browser.

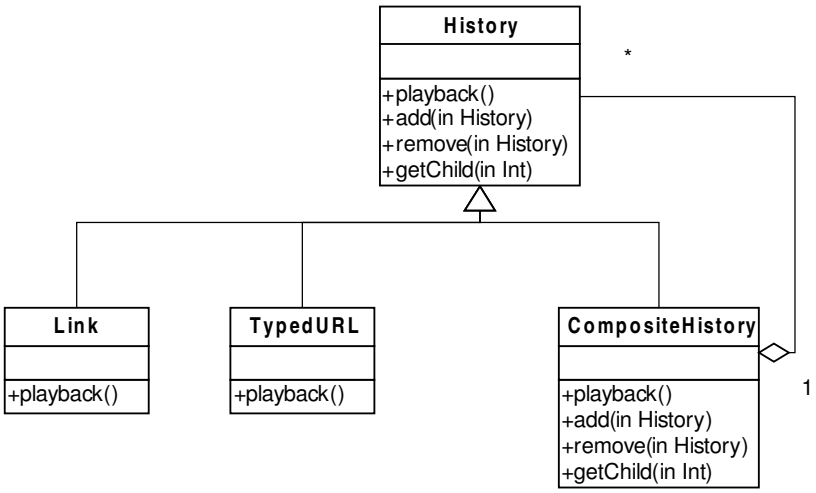

Figura 15: Estrutura do Playback

5.1.4.2.Comentário

A funcionalidade de comentário é implementada através de um campo *Coment* presente na classe *History.* Desta forma, para adicionar um comentário basta chamar o método *setComent(String)* da classe *History*. No momento do playback está informação será passada para a camada Cliente, que tornará o comentário disponível para o usuário.

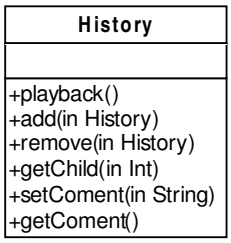

Figura 16: Comentários no playback

5.1.5.Camada de Dados

Esta camada é responsável pela persistência dos históricos gerados pelo Timeline Web Browser. As informações serão persistidas em arquivos XML, facilitando a transferência destas informações através dos web services presentes na camada Web.

Cada classe que representa um objeto do histórico, ou seja, que representa uma ação, tem conhecimento de como gerar o XML responsável por persistir as informações relevantes presentes em um objeto da classe. Para isso, o método abstrato *generateXML* de *History* deve ser implementado por cada uma de suas subclasses concretas, como mostra a Figura 17. Então, estes XMLs são persistidos em arquivos através da API de manipulação de arquivos de Java, *java.io*. Para persistir um histórico, basta passar um objeto do tipo *History* para o método *generateXML* da classe *timelinewebbrowser.util.XMLGenerator*.

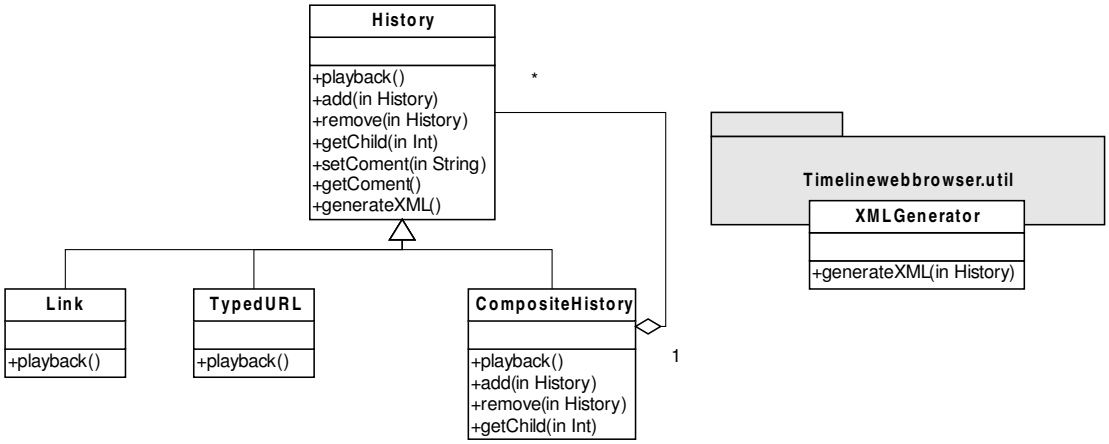

Figura 17: Estrutura da persistência de dados

Além de persistir os dados, esta camada é responsável também por ler os dados dos arquivos XML já salvos para serem usados, por exemplo, em um playback. Para isso é usada a API de SAX (*Simple API for XML*) , que possui um conjunto de funções para lidar com arquivos XML. A função para carregar estes arquivos é chamada de *parseXML*, ela recebe como parâmetros dois objetos *java.util.Date*, que representam o intervalo de tempo usado na busca pelos históricos e retorna um objeto *History*. Esta função é implementada em uma classe chamada *timelinewebbrowser.util.XMLParser*.

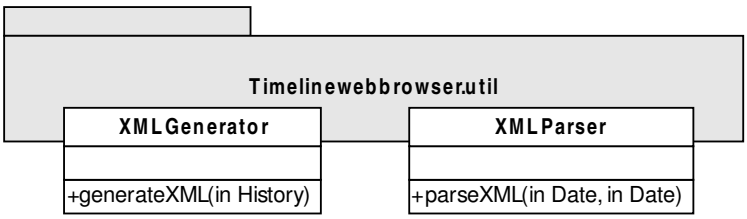

Figura 18: Camada de dados

Conclusões

"Plaudite, amici, comedia finita est."
Ludwig van Beethoven

Este trabalho realizou uma análise das descrições de Web 2.0, com foco no conceito de plataforma web, estabelecendo uma arquitetura para aplicações baseadas neste conceito. Também desenvolveu uma aplicação piloto baseada nesta arquitetura chamada de Timeline Web Browser, que semelhante ao del.icio.us ou shadows , serve para compartilhar históricos na web, aos invés de favoritos, como é o caso das duas aplicações citadas acima.

Inicialmente foi dada uma visão geral de Web 2.0 e sobre os vários conceitos associados a este termo. O trabalho segue apresentando com mais detalhes o conceito de plataforma web e em seguida descreve as tecnologias mais utilizadas para desenvolver aplicações web hoje em dia. Em seguida estabelece uma arquitetura padrão baseada nestas tecnologias e então descreve o Timeline Web Browser, uma aplicação piloto para esta arquitetura.

No decorrer deste trabalho são explicadas com um bom nível de detalhes as principais tecnologias que contribuem para a plataforma web, como: SOA, web service, XML, SOAP, UDDI, entre outras. Isso serve como uma preparação para a modelagem de uma arquitetura que servirá como base para o desenvolvimento de aplicações na plataforma web.

As grandes contribuições deste trabalho foram: a modelagem de uma arquitetura em camadas, que pode ser usada para o desenvolvimento de aplicações na plataforma web, baseando-se nas tecnologias mais utlizadas para desenvolvimento deste tipo de aplicação; e o desenvolvimento inicial de um sistema baseado nesta arquitetura chamado de Timeline Web Browser, uma aplicação que fornece serviços para o compartilhamento de histórico de navegação na internet, permitindo a edição destes históricos e a busca de informações por data.

5.2.Trabalhos futuros

Este trabalho pode dar origem a diversos outros trabalhos, tanto em relação à arquitetura estabelecida quanto em relação ao Timeline Web Browser.

Em relação à arquitetura, o trabalho mais importante é aplicar esta arquitetura nas mais diversas aplicações baseadas na plataforma web e medir sua adequação a estas aplicações, e se necessário realizar ajustes a arquitetura estabelecida.

Em relação ao Timeline Web Browser, várias funcionalidades podem ainda precisam ser implementadas:

- Implementar a *engine* de busca do timeline web browser.
- Adicionar eventos de gravação a aplicação.
- Implementar outras aplicações fazendo uso dos serviços disponibilizados pela API.

5.3. Considerações finais

Neste capítulo foram apresentadas as contribuições deste trabalho e possíveis trabalhos futuros. Web 2.0 é um ponto chave para este trabalho, bem como uma de suas facetas mais importantes, a de plataforma web.

Apêndice A

Este apêndice possui uma listagem de um documento WSDL para um serviço de busca de livros. Este serviço possui uma operação que recebe o ISBN do livro e retorna como resultado o título do livro.

```
/**
 * Copyright (c) 2005 Sun Microsystems, Inc. All Rights Reserved.
 *
 * This software is the confidential and proprietary information of
Sun
 * Microsystems, Inc. ("Confidential Information").  You shall not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered
into
 * with Sun.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY
OF THE
 * SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
 * PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR ANY
DAMAGES
 * SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 */

<?xml version="1.0" encoding="UTF-8"?>
<definitions name="BookSearch"
             targetNamespace="http://myexample.com/booksearch.wsdl"
             xmlns:tns="http://myexample.com/booksearch.wsdl"
             xmlns:xsd1="http://myexample.com/booksearch.xsd"

         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
       <schema targetNamespace="http://myexample.com/booksearch.xsd"
       <element name="BookRequest">
           <complexType>
           <all>
              <element name="isbn"
                    type="string"/>
           </all>
           </complexType>
       </element>
       <element name="BookResponse">
           <complexType>
           <all>
              <element name="title"
                    type="string"/>
           </all>
           </complexType>
       </schema>
    </types>
    <message name="BookSearchInput">
        <part name="body" element:"xsdl:BookRequest"/>
    </message>
```

```
    <message name="BookSearchOutput">
       <part name="body" type="xsdl:BookResponse"/>
    </message>
    <portType name="BooksPortType">
        <operation name="getBooks"
            <input message="tns:BookRequest"/>
            <output
               message="tns:BookResponse"/>
        </operation>
    </portType>
    <binding name="BooksSearchBinding"
                                type="tns:BooksPortType">
        <operation name="BookRequest">
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal""/>
            </output>
            <soap:operation soapAction=""/>
        </operation>
        <soap:binding
                transport="transport=http://schemas.xmlsoap.org/soap/
http"
                style="rpc"/>
    </binding>
    <service name="BookSearchService">
        <port name="BooksSearchPort"
                             binding="tns:BooksSearchBinding">
        <soap:address
                location="http://example.com/booksearch"/>
        </port>
    </service>
</definitions>
```

Referências

[1] O'REILLY, Tim. What is Web 2.0: Design patterns and business models for the next generation of software. Disponível em: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1. Acessado em: Agosto de 2006.

[2] LAMONICA, Martin. From web page to web platform. Disponível em: http://news.com.com/From+Web+page+to+Web+platform+-+page+2/2100-7345_3-5833940-2.html?tag=st.next. Acessado em: Agosto de 2006.

[3] HINCHCLIFFE, Dion. The SOA with reach: Web-Oriented Architecture. Disponível em: http://blogs.zdnet.com/Hinchcliffe/?p=27. Acessado em: Junho de 2006.

[4] ORT, Ed. Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools. Disponível em: http://java.sun.com/developer/technicalArticles/WebServices/soa2/. Acessado em: Agosto de 2006.

[5] CHRISTENSEN, Erik; CURBERA, Francisco; MEREDITH, Greg; WEERAWARANA, Sanjiva. Web Services Description Language (WSDL) 1.1. Disponível em: http://www.w3.org/TR/wsdl. Acessado em: Agosto de 2006.

[6] Google Maps. Disponível em: http://maps.google.com. Acessado em: Setembro de 2006.

[7] Del.icio.us. Disponível em: http://del.icio.us. Acessado em: Agosto de 2006.

[8] Digg. Disponível em: http://www.digg.com. Acessado em: Agosto de 2006.

[9] Flickr. Disponível em: http://www.flickr.com. Acessado em: Agosto de 2006.

[10] Youtube. Disponível em: http://www.youtube.com. Acessado em: Agosto de 2006.

[11] ThinkFree. Disponível em: http://www.thinkfree.com. Acessado em: Setembro de 2006.

[12] Writely. Disponível em: http://www.writely.com. Acessado em: Setembro de 2006.

[13] EyeOS. Disponível em: http://www.eyeos.org. Acessado em: Setembro de 2006.

[14] NetVibes. Disponível em: www.netvibes.com. Acessado em: Setembro de 2006.

[15] O'REILLY, Tim. The Architecture of Participation. Disponível em: http://www.oreillynet.com/pub/wlg/3017. Acessado em: Agosto de 2006.

[16] BUSCHMANN, Frank. Pattern-Oriented Software Architecture, Volume 1: A system of Patterns. Jon Wiley & Sons, 1996.

[17] MONDAY, Paul. Web Service Patterns: Java Edition. A! Apress, 2003.

[18] Apache Axis. Disponível em: http://xml.apache.org/axis. Acessado em: Setembro de 2006.

[19] FOWLER, Martin. Patterns of Enterprise Application Architecture. Addison-Wesley, 2005.

[20] Shadows. Disponível em: http://www.shadows.com. Acessado em Setembro de 2006.

[21] SWT. Disponível em: http://www.eclipse.org. Acessado em: Agosto de 2006.

[22] GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos. Bookman, 2000.

[23] SAX (Simple API for XML). Disponível em: http://www.saxproject.org/. Acessado em: Agosto de 2006.